davidg

Cotxox

> Vamos a construir un prototipo de la aplicación Cotxox
> que permite gestionar el traslado de turistas desde el aeropuerto
> a determinadas zonas de la isla de Mallorca, y provoca huelgas
> de taxistas a discreción.

Prepara el proyecto

a. Crea un nuevo proyecto en tu cuenta en **Github**
b. Crea un nuevo directorio en tu equipo y **clona el repositorio** de Github.
c. Abre Eclipse /Netbeans y **establece como workspace** el directorio que has clonado.
d. Crea un proyecto con tu **nombre y apellidos**.
e. Copia y pega la función principal `Cotxox.java`. Utilízala como guía en el proceso TDD. **No puedes modificar su código**.
f. Completa las clases que necesites **implementando los casos test que necesites**.
g. Organiza las clases en sus paquetes correpondientes. Observa el esquema propuesto en el `import` de la clase principal `Cotxox.java`

```
```
import org.foobarspam.cotxox.carrera.Carrera;
import org.foobarspam.cotxox.conductores.Conductor;
import org.foobarspam.cotxox.conductores.PoolConductores;
import org.foobarspam.cotxox.tarifa.Tarifa;
```
```

Cómo entregar el código

a. Accede al workspace de Eclipse y busca la carpeta que tiene igual nombre que tu proyecto.
b. Comprime esa carpeta.
c. Envíame el archivo por correo electrónico.

d. **Realiza commits periódicamente** mientras avanzas en el desarrollo de la aplicación.

Salida de la aplicación

Intenta que la salida del programa sea lo más parecida posible a la imagen que se proporciona:

https://foobarspam.slack.com/files/davidg/F46ERLXN0/salida_consola_cotxox.png

Historias de usuario

Crea un documento donde escribas las historias de usuario correspondientes a los 5 hitos del proceso,
representados por estas 5 imágenes:

https://foobarspam.slack.com/files/davidg/F46F15N0M/01_pedir_un_cotxo.png

https://foobarspam.slack.com/files/davidg/F46ER8UVA/02_cu__nto_va_a_costarte.png

https://foobarspam.slack.com/files/davidg/F45M21FT2/03_informaci__n_de_tu_cotxo_y_conductor.png

https://foobarspam.slack.com/files/davidg/F45MVNRBK/04_paga_en_la_app.png

https://foobarspam.slack.com/files/davidg/F4730S3FG/05_valora_a_la_conductora.png

Diagrama de clases UML

Realiza a mano alzada un pequeño diagrama de clases UML que muestre la relación entre las clases que has construido, con su interfaz pública y privada.

Código

Clase Carrera

ATRIBUTOS

- crea las variables de instancia que estimes oportunas.
- `tiempoEsperado` del trayecto
- `tiempoCarrera` tiempo real empleado en el trayecto
- `costeTotal` real del trayecto
- `conductor` asignado a la carrera

MÉTODOS

- `getTarjetaCredito()` devuelve el número de la tarjeta de crédito del usuario/a.
- `getOrigen()` devuelve el lugar de origen del trayecto.
- `getDestino()` devuelve el lugar de destino del trayecto.
- `getDistancia()` devuelve la distancia entre el origen y el destino.
- `getCosteEsperado()` devuelve el coste esperado del trayecto, cuyo cálculo es responsabilidad de la clase `Tarifa`.
- `asignarConductor(PoolConductores conductores)` recibe la flota de conductores y asigna un conductor a la carrera. Le pide a la clase `PoolConductores` que le asigne un `conductor`.
- `getCosteEsperado()` le pregunta a la clase `Tarifa` cuál es el coste total esperado.
- `realizarPago(pago)` recibe el pago y lo almacena en el atributo `costeTotal`
- `recibirPropina(propina)` recibe la propina que paga el usuario
- `liberarConductor()` establece que el conductor asignado a la carrera queda libre tras el servicio.

Clase Tarifa

ATRIBUTOS

  - `id` = identificador de la bicicleta: un número de tres dígitos.
  - `costeMilla` Cotxox fija en 1.35€ el coste de la milla.

- `costeMinuto` Cotxox fija en 0.35€ el coste del minuto.
- `costeMinimo` El coste mínimo que cobra Cotxox por un viaje es de 5€.
- `porcentajeComision` la comisión que cobra cotxox sobre el coste del viaje es del 20%.

MÉTODOS

- `getCosteDistancia(distancia)` devuelve la parte del coste del trayecto debido al a distancia.
- `getCosteTiempo(minutos)` devuelve la parte del coste del trayecto debido a su duración en minutos.
- `getCosteTotalEsperado(carrera)` devuelve el coste total esperado de la `carrera` que recibe en función de la distancia esperada

y el tiempo esperado. El coste total esperado no puede ser inferior al mínimo.

Clase Conductor

ATRIBUTOS

- `nombre` del conductor
- `modelo` modelo del coche
- `matricula`
- `valoracionMedia` valoración media del conductor
- `valoraciones` array de longitud variable que almacena todas las valoraciones del conductor
- `ocupado` indica si el conductor está prestando un servicio o está libre.

MÉTODOS

- `setValoracion(valoracion)` añade la nueva valoración y actualiza la valoración media del conductor.

Clase PoolConductores

ATRIBUTOS

- `poolConductores` es un array de longitud variable de `conductor` es

MÉTODOS

- El constructor `PoolConductores(conductores)` recibe un array de longitud variable de conductores.
- `asignarConductor()` selecciona un conductor **libre** entre la flota y lo devuelve, estableciendo que ese conductor está ahora **ocupado**.